```
	/*Note that these include files that are for the purposes of giving
		graphical output (ccc_win.cpp) and conforming to the ANSI C++
		standard (ccc_ansi.cpp).  These include files that are inessential
		to the optimization algorithm.  they are copyrighted by Cay
		Horstmann, and are available at http://www.horstmann.com
	*/

#include "ccc_win.cpp"
#include "ccc_ansi.cpp"

int totalbits(double base, double rho, double absorption, double rippletime);

int main(){

	double bits = cwin.get_double("Enter length of adder operands in bits:
");
	double rho = cwin.get_double("Enter ratio between skip-time and ripple-
time: ");
	double absorption = cwin.get_double("Enter constant carry absorption
delay (try 0 if unknown): ");
	double rippletime = cwin.get_double("Enter ripple time (try 1 if
unknown): ");

	/*  Find first approximate fitting triangle by computing
		a triangle such that 0.5*base*height = # bits.
		Base on this approximation we must have
		0.5*rho*base^2 = # bits.
	*/

	double base = 0;
	base = sqrt(2.0*bits/rho);
	double height = 0;
	height = rho*base;

	//Increment base if necessary until the triangle
	//contains enough bits.
	while (totalbits(base, rho, absorption, rippletime) < bits)
		base = base + 1.0;

	//Now do a binary search to find tight-fitting height.
	double high = base, low = 0.0;

	const double tolerance = 1.0e-7;

	while (fabs(high-low) > tolerance){
		base = (high+low)/2.0;
		if (totalbits(base, rho, absorption, rippletime) < bits)
			low = base;
		else
			high = base;
	} //while

	//At this point, high and low are within tolerance of
	//each other.  High is guaranteed to be a large enough base
	//to contain all bits.  So set:
	base = high;
	bits = totalbits(high, rho, absorption, rippletime);

	//Plot the answer:
```

```
        int minx = - static_cast<int>(0.2* base);
        if (minx > -2) minx = -2;
        int maxx = static_cast<int>(1.2* base);
        int miny = - static_cast<int>(0.2 * base * rho);
        if (miny > -2) miny = -2;
        int maxy = static_cast<int>(1.2* (base * rho + absorption/rippletime));
        cwin.coord(minx, maxy, maxx, miny);
        Line xaxis (Point(minx, 0), Point(maxx, 0));
        Line yaxis (Point(0, miny), Point(0, maxy));
        Line graph (Point(base, 0), Point(0, base * rho));

        Message announce_nbits (Point(0.6 * base, 0.8*base*rho),
              "Bits:");
        Message number_bits (Point(0.85 * base, 0.8*base*rho), bits);

        Message announce_delay (Point(0.6 * base, 0.6*base*rho),
              "Carry delay:");
        Message delay (Point(0.85*base, 0.6*base*rho),
              absorption + rippletime * (base * rho - 1.0));

        cwin << xaxis << yaxis << graph
              << announce_nbits << number_bits
              << announce_delay << delay;

        int ibase = static_cast<int> (base);
        for (int i = ibase; i >= 0; i--){
              double height = (base - i)*rho;
              int iheight = -1;//floor of height
              if (i == 0)
                    iheight = static_cast<int>(height+(absorption/rippletime));
              else
                    iheight = static_cast<int>(height);
              if (iheight > 0){
                    cwin << Point(i, iheight);
                    cwin << Message(Point(i, iheight), iheight);
              }
        } //for
        return 0;
} //main()

/*      Routine to find total number of bits contained in
        a triangle with a specified base length,
        slope absolute value, and absorption time.
*/
int totalbits(double base, double rho, double absorption, double rippletime){
        int ibase = static_cast<int> (base);
        int bits = 0;
        for (int i = ibase; i >= 0; i--){
              double height = (base - i)*rho;
              int iheight = -1; //floor of height
              if (i == 0)
                    iheight = static_cast<int>(height+(absorption/rippletime));
              else
                    iheight = static_cast<int>(height);
              bits += iheight;
        } //for

        return bits;
} //totalbits()
```

```
#!/usr/bin/perl
# Perlscript for generating one-level carry-skip adders with carry strength

# usage csagen.pl outfile lsblk_size blk1size blk2size ... msblk_size
$numargs = @ARGV;

if ($numargs < 2){ die "Usage csagen.pl outfile lsblk_size blk1size blk2size
... msblk_size\n" }

$outfile = shift(@ARGV);

open OUT, ">$outfile";

print OUT '-- Fast, small, and low-power carry-skip adder with carry strength
following', "\n";
print OUT '-- designs by Vitit Kantabutra, Pasquale Corsonello, and Stefania
Perri', "\n";
print OUT '-- Perl and VHDL code (c) Vitit Kantabutra, 2001', "\n";
print OUT '-- Generated by the Perlscript csagen.pl', "\n\n";

# generate entity-architecture pair for adder bits 0 and 1, and also
# for the higher bits
&add_01_gen;
&add_hibits_gen;
# -----------------------------------

# Generate all the blocks required.  Don't repeat a size!
# Each block has no carry strength output, but instead has a cout select mux.

# Since this program is meant for optimum adders, the least significant
# block should be small.  Thus there is no need for a least significant
# block without carry strength computation.

foreach $size (@ARGV){
  unless (&member($size, @sizes_generated)){
    &blk_gen($size);
    push(@sizes_generated, $size);
  } # unless
} # foreach

# -----------------------------------

# Generate entire adder, which is just a simple series of port map

# CODE HERE FOR GENERATING ENTIRE ADDER
$totalsize = &sum(@ARGV);
$totalsize_minus_1 = $totalsize - 1;

print OUT '-- TOP LEVEL ENTITY-ARCHITECTURE PAIR',"\n\n";
# output the library and use lines
print OUT "library IEEE;\nuse IEEE.std_logic_1164.all;\n\n";

# Output the ENTITY
print OUT "entity add$totalsize is\n";
print OUT "  port (\n";
print OUT "     x, y: in STD_LOGIC_VECTOR (0 to $totalsize_minus_1);\n";
print OUT "     cin: in STD_LOGIC;\n";
print OUT "     s: out STD_LOGIC_VECTOR (0 to $totalsize_minus_1);\n";
print OUT "     cout: out STD_LOGIC\n";
```

```
print OUT "  );\n";
print OUT "end add$totalsize;\n\n";

# ----------------------------------
#ARCHITECTURE PART OF THE ENTIRE ADDER

$numblocks = $numargs-1;
print "The blocks are @ARGV\n";
print "Number of blocks is $numblocks\n";
print OUT "architecture add$totalsize\_arch of add$totalsize is\n";

$numblocks_minus_1 = $numblocks - 1;
if ($numblocks > 1) {
      print OUT "  signal c: STD_LOGIC_VECTOR (1 to  $numblocks_minus_1);\n"
};

foreach $size (@ARGV){
  unless (&member($size, @sizes_generated2)){
    &component_blk_gen($size);
    push(@sizes_generated2, $size);
  } # unless
} # foreach

# ----------------------------------
# begin-end pair of the ARCHITECTURE
print OUT "begin\n";
$currentlsb = 0;
$currentblk = 0;
foreach $size (@ARGV){
  print OUT "  blk$currentblk: add$size port map\n";
  $currentmsb = $currentlsb + $size - 1;
  if ($currentmsb > $currentlsb){
    print OUT "    (x => x($currentlsb to $currentmsb),\n";
    print OUT "     y => y($currentlsb to $currentmsb),\n";
    print OUT "     s => s($currentlsb to $currentmsb),\n";
  } else {
    print OUT "    (x => x($currentlsb),\n";
    print OUT "     y => y($currentlsb),\n";
    print OUT "     s => s($currentlsb),\n";
  }
  if ($currentlsb == 0) {
    print OUT "     cin => cin,\n";
   } else {
     print OUT "     cin => c($currentblk),\n";
   }
  if ($currentblk == $numblocks-1) {
     print OUT "     cout => cout\n";
   } else {
     $nextblk = $currentblk+1;
     print OUT "     cout => c($nextblk)\n";
   }
  print OUT "  );\n";

  $currentlsb += $size;
  $currentblk++;
}  # foreach
print OUT "end add$totalsize\_arch;\n";

close OUT;

# ----------------------------------
# Subroutine for generating 1-bit adders
sub add_01_gen {
```

```
  open IN, "add_01.vhdl";
  while (<IN>){ print OUT; }
  close IN;
  print OUT "\n";
} # sub add_01_gen

sub add_hibits_gen {
  open IN, "add_hibits.vhdl";
  while (<IN>){ print OUT; }
  close IN;
  print OUT "\n";
} # sub add_hibits_gen

# -----------------------------------
# subroutine to figure out whether a NUMBER (first argument) is in
# the list consisting of the other args

# for strings change == to eq

sub member {
  $retval = '';
  @the_list = @_; shift @the_list;
  foreach (@the_list){
    if (@_[0] == $_){
      $retval = 'true';
    } # if
  } # foreach
  $retval;
} # sub member for numbers

# -----------------------------------
# subroutine to generate an adder block
sub blk_gen {
  my($size) = @_[0];
  if ($size == 1){
    open IN, "blk1.vhdl";
    while (<IN>){ print OUT; }
    close IN;
  } elsif ($size == 2){
    open IN, "blk2.vhdl";
    while (<IN>){ print OUT; }
    close IN;
  } else { # block has more than 2 bits
    $size_minus_1 = $size - 1;

    # Output the library inclusion and the ENTITY part of the VHDL code

    print OUT "library IEEE;\n";
    print OUT "use IEEE.std_logic_1164.all;\n\n";
    print OUT "entity add$size is\n";
    print OUT "  port(\n";
    print OUT "     x: in STD_LOGIC_VECTOR(0 to $size_minus_1);\n";
    print OUT "     y: in STD_LOGIC_VECTOR(0 to $size_minus_1);\n";
    print OUT "     cin: in STD_LOGIC;\n";
    print OUT "     cout: out STD_LOGIC;\n";
    print OUT "     s: out STD_LOGIC_VECTOR(0 to $size_minus_1)\n";
    print OUT "  );\n";
    print OUT "end add$size;\n\n";

    # Output the ARCHITECTURE part of the VHDL code, which contains
    #    SIGNAL declarations
    #    COMPONENT declarations
```

```
    #    begin-end pair, with contents consisting mostly of component
instantiations and signal assignments

    print OUT "architecture add$size\_arch of add$size is\n";
    print OUT "  signal xn: STD_LOGIC_VECTOR(1 to $size_minus_1);\n";
    print OUT "  signal c, cs: STD_LOGIC_VECTOR(1 to $size);\n";

    # component declarations
    open IN, "addcomp.vhdl";
    while (<IN>){ print OUT; }
    close IN;

    # begin line
    print OUT "begin\n";
    # Component instantiation: bits 0 and 1

    open IN, "add_01_instantiation.vhdl";
    while (<IN>){ print OUT; }
    close IN;

    # Component instantiation: higher bits
    print OUT "\n-- component instantiation of higher bits\n\n";
    my($i);
    for ($i = 2; $i < $size; $i++){
      $i_plus_1 = $i + 1;
      print OUT "  addbit$i: add_hibits port map\n";
      print OUT "    (x => x($i), y => y($i), blkcin => cin, ripcin => c($i),
\n";
      print OUT "     csin => cs($i), s => s($i), cout => c($i_plus_1), csout
=> cs($i_plus_1) );\n";
    } # for
    print OUT "  cout <= c($size) when cs($size) = '1' else cin;\n";
  print OUT "end add$size\_arch;\n\n";
  } # else block has more than 2 bits
} # sub blk_gen

# ---------------------------------
# subroutine to generate an adder block as component

sub component_blk_gen {
  my($size) = @_[0];
  if ($size == 1){
    print OUT "  component add1\n";
    print OUT "    port(\n";
    print OUT "      x: in STD_LOGIC;\n";
    print OUT "      y: in STD_LOGIC;\n";
    print OUT "      cin: in STD_LOGIC;\n";
    print OUT "      cout: out STD_LOGIC;\n";
    print OUT "      s: out STD_LOGIC\n";
    print OUT "    );\n";
    print OUT "  end component;\n\n";
  } else { # block has more than 1 bit
    $size_minus_1 = $size - 1;

    print OUT "  component add$size\n";
    print OUT "    port(\n";
    print OUT "      x: in STD_LOGIC_VECTOR(0 to $size_minus_1);\n";
    print OUT "      y: in STD_LOGIC_VECTOR(0 to $size_minus_1);\n";
    print OUT "      cin: in STD_LOGIC;\n";
    print OUT "      cout: out STD_LOGIC;\n";
    print OUT "      s: out STD_LOGIC_VECTOR(0 to $size_minus_1)\n";
    print OUT "    );\n";
    print OUT "  end component;\n\n";
```

```
  } # if-else
} # sub component_blk_gen

# ------------------------------------------------
# subroutine that sums all the numbers in an array
sub sum {
  my($total) = 0;
  foreach $number (@_){
    $total += $number;
  } # foreach
  $total;
} # sum
```

```
-- This part read from a file called add_01.vhdl
-- One bit adder for bit 0 (lsb) or bit 1

library IEEE;
use IEEE.std_logic_1164.all;

entity add_01 is
  port (
    x: in STD_LOGIC;
    y: in STD_LOGIC;
    cin: in STD_LOGIC;
    s: out STD_LOGIC;
    cout: out STD_LOGIC;
    x_xnor_y: out STD_LOGIC
  );
end add_01;

architecture add_01_arch of add_01 is
  signal x_xnor_y_signal: STD_LOGIC;
begin
  x_xnor_y_signal <= x xnor y;
  x_xnor_y <= x_xnor_y_signal;
  s <= cin xnor x_xnor_y_signal;
  cout <= x when x_xnor_y_signal = '1' else
          cin;
end add_01_arch;
```

```
-- This part read from a file called add_01_instantiation.vhdl
-- component instantiation of adder bits 0 and 1 for block of bits

 addbit0: add_01 port map
   (x => x(0),
    y => y(0),
    cin => cin,
    s => s(0),
    cout => c(1),
    x_xnor_y => cs(1)
   );

 addbit1: add_01 port map
   (x => x(1),
    y => y(1),
    cin => c(1),
    s => s(1),
    cout => c(2),
    x_xnor_y => xn(1)
   );

 cs(2) <= xn(1) or cs(1);
```

```
-- This part read from a file called add_hibits.vhdl
-- One-bit adder, not lsb (bit 0) or bit 1

library IEEE;
use IEEE.std_logic_1164.all;

entity add_hibits is
  port (
    x: in STD_LOGIC;
    y: in STD_LOGIC;
    blkcin: in STD_LOGIC;
    ripcin: in STD_LOGIC;
    csin: in STD_LOGIC;
    s: out STD_LOGIC;
    cout: out STD_LOGIC;
    csout: out STD_LOGIC
  );
end add_hibits;

architecture add_hibits_arch of add_hibits is
  signal cc, x_xnor_y : STD_LOGIC;

begin
  x_xnor_y <= x xnor y;
  csout <= csin or x_xnor_y;
  cout <= x when x_xnor_y = '1' else
          ripcin;
  cc <= ripcin when csin = '1' else
        blkcin;
  s <= cc xnor x_xnor_y;
end add_hibits_arch;
```

```
component add_01
   port (
      x: in STD_LOGIC;
      y: in STD_LOGIC;
      cin: in STD_LOGIC;
      s: out STD_LOGIC;
      cout: out STD_LOGIC;
      x_xnor_y: out STD_LOGIC
   );
end component;
component add_hibits
  port (
      x: in STD_LOGIC;
      y: in STD_LOGIC;
      blkcin: in STD_LOGIC;
      ripcin: in STD_LOGIC;
      csin: in STD_LOGIC;
      s: out STD_LOGIC;
      cout: out STD_LOGIC;
      csout: out STD_LOGIC
  );
end component;
```

```
--This part read from a file called blk1.vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity add1 is
  port(
    x: in STD_LOGIC;
    y: in STD_LOGIC;
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC
  );
end add1;

architecture add1_arch of add1 is
  component add_01
     port (
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        cin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        x_xnor_y: out STD_LOGIC
     );
  end component;
begin
-- component instantiation of adder bit 0
  addbit0: add_01 port map
    (x => x,
     y => y,
     cin => cin,
     s => s,
     cout => cout
    );
end add1_arch;
```

```
-- This part read from a file called blk2.vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity add2 is
  port(
    x: in STD_LOGIC_VECTOR(0 to 1);
    y: in STD_LOGIC_VECTOR(0 to 1);
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC_VECTOR(0 to 1)
  );
end add2;

architecture add2_arch of add2 is
  signal c, cs: STD_LOGIC_VECTOR(1 to 2);
  signal xn: STD_LOGIC;
  component add_01
    port (
      x: in STD_LOGIC;
      y: in STD_LOGIC;
      cin: in STD_LOGIC;
      s: out STD_LOGIC;
      cout: out STD_LOGIC;
      x_xnor_y: out STD_LOGIC
    );
  end component;
begin
  addbit0: add_01 port map
    (x => x(0),
     y => y(0),
     cin => cin,
     s => s(0),
     cout => c(1),
     x_xnor_y => cs(1)
    );

  addbit1: add_01 port map
    (x => x(1),
     y => y(1),
     cin => c(1),
     s => s(1),
     cout => c(2),
     x_xnor_y => xn
    );
    cs(2) <= xn or cs(1);
    cout <= c(2) when cs(2) = '1' else cin;
end add2_arch;
```

```
-- Fast, small, and low-power carry-skip adder with carry strength following
-- designs by Vitit Kantabutra, Pasquale Corsonello, and Stefania Perri
-- Perl and VHDL code (c) Vitit Kantabutra, 2001
-- Generated by the Perlscript csagen.pl

-- This part read from a file called add_01.vhdl
-- One bit adder for bit 0 (lsb) or bit 1

library IEEE;
use IEEE.std_logic_1164.all;

entity add_01 is
  port (
    x: in STD_LOGIC;
    y: in STD_LOGIC;
    cin: in STD_LOGIC;
    s: out STD_LOGIC;
    cout: out STD_LOGIC;
    x_xnor_y: out STD_LOGIC
  );
end add_01;

architecture add_01_arch of add_01 is
  signal x_xnor_y_signal: STD_LOGIC;
begin
  x_xnor_y_signal <= x xnor y;
  x_xnor_y <= x_xnor_y_signal;
  s <= cin xnor x_xnor_y_signal;
  cout <= x when x_xnor_y_signal = '1' else
          cin;
end add_01_arch;

-- This part read from a file called add_hibits.vhdl
-- One-bit adder, not lsb (bit 0) or bit 1

library IEEE;
use IEEE.std_logic_1164.all;

entity add_hibits is
  port (
    x: in STD_LOGIC;
    y: in STD_LOGIC;
    blkcin: in STD_LOGIC;
    ripcin: in STD_LOGIC;
    csin: in STD_LOGIC;
    s: out STD_LOGIC;
    cout: out STD_LOGIC;
    csout: out STD_LOGIC
  );
end add_hibits;

architecture add_hibits_arch of add_hibits is
  signal cc, x_xnor_y : STD_LOGIC;

begin
  x_xnor_y <= x xnor y;
  csout <= csin or x_xnor_y;
  cout <= x when x_xnor_y = '1' else
          ripcin;
  cc <= ripcin when csin = '1' else
        blkcin;
```

```
  s <= cc xnor x_xnor_y;
end add_hibits_arch;


--This part read from a file called blk1.vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity add1 is
  port(
    x: in STD_LOGIC;
    y: in STD_LOGIC;
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC
  );
end add1;

architecture add1_arch of add1 is
  component add_01
     port (
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        cin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        x_xnor_y: out STD_LOGIC
     );
  end component;
begin
-- component instantiation of adder bit 0
  addbit0: add_01 port map
    (x => x,
     y => y,
     cin => cin,
     s => s,
     cout => cout
    );
end add1_arch;

-- This part read from a file called blk2.vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity add2 is
  port(
    x: in STD_LOGIC_VECTOR(0 to 1);
    y: in STD_LOGIC_VECTOR(0 to 1);
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC_VECTOR(0 to 1)
  );
end add2;

architecture add2_arch of add2 is
  signal c, cs: STD_LOGIC_VECTOR(1 to 2);
  signal xn: STD_LOGIC;
  component add_01
    port (
      x: in STD_LOGIC;
      y: in STD_LOGIC;
      cin: in STD_LOGIC;
      s: out STD_LOGIC;
```

```
      cout: out STD_LOGIC;
      x_xnor_y: out STD_LOGIC
    );
  end component;
begin
  addbit0: add_01 port map
    (x => x(0),
     y => y(0),
     cin => cin,
     s => s(0),
     cout => c(1),
     x_xnor_y => cs(1)
    );

  addbit1: add_01 port map
    (x => x(1),
     y => y(1),
     cin => c(1),
     s => s(1),
     cout => c(2),
     x_xnor_y => xn
    );
    cs(2) <= xn or cs(1);
    cout <= c(2) when cs(2) = '1' else cin;
end add2_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity add3 is
  port(
    x: in STD_LOGIC_VECTOR(0 to 2);
    y: in STD_LOGIC_VECTOR(0 to 2);
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC_VECTOR(0 to 2)
  );
end add3;

architecture add3_arch of add3 is
  signal xn: STD_LOGIC_VECTOR(1 to 2);
  signal c, cs: STD_LOGIC_VECTOR(1 to 3);
  component add_01
     port (
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        cin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        x_xnor_y: out STD_LOGIC
     );
  end component;
  component add_hibits
    port (
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        blkcin: in STD_LOGIC;
        ripcin: in STD_LOGIC;
        csin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        csout: out STD_LOGIC
    );
```

```
  end component;
begin
-- This part read from a file called add_01_instantiation.vhdl
-- component instantiation of adder bits 0 and 1 for block of bits

  addbit0: add_01 port map
    (x => x(0),
     y => y(0),
     cin => cin,
     s => s(0),
     cout => c(1),
     x_xnor_y => cs(1)
    );

  addbit1: add_01 port map
    (x => x(1),
     y => y(1),
     cin => c(1),
     s => s(1),
     cout => c(2),
     x_xnor_y => xn(1)
    );

  cs(2) <= xn(1) or cs(1);


-- component instantiation of higher bits

  addbit2: add_hibits port map
    (x => x(2), y => y(2), blkcin => cin, ripcin => c(2),
     csin => cs(2), s => s(2), cout => c(3), csout => cs(3) );
  cout <= c(3) when cs(3) = '1' else cin;
end add3_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity add4 is
  port(
    x: in STD_LOGIC_VECTOR(0 to 3);
    y: in STD_LOGIC_VECTOR(0 to 3);
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC_VECTOR(0 to 3)
  );
end add4;

architecture add4_arch of add4 is
  signal xn: STD_LOGIC_VECTOR(1 to 3);
  signal c, cs: STD_LOGIC_VECTOR(1 to 4);
  component add_01
     port (
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        cin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        x_xnor_y: out STD_LOGIC
     );
  end component;
  component add_hibits
    port (
        x: in STD_LOGIC;
```

```
        y: in STD_LOGIC;
        blkcin: in STD_LOGIC;
        ripcin: in STD_LOGIC;
        csin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        csout: out STD_LOGIC
    );
  end component;
begin
-- This part read from a file called add_01_instantiation.vhdl
-- component instantiation of adder bits 0 and 1 for block of bits

  addbit0: add_01 port map
    (x => x(0),
     y => y(0),
     cin => cin,
     s => s(0),
     cout => c(1),
     x_xnor_y => cs(1)
    );

  addbit1: add_01 port map
    (x => x(1),
     y => y(1),
     cin => c(1),
     s => s(1),
     cout => c(2),
     x_xnor_y => xn(1)
    );

  cs(2) <= xn(1) or cs(1);


-- component instantiation of higher bits

  addbit2: add_hibits port map
    (x => x(2), y => y(2), blkcin => cin, ripcin => c(2),
     csin => cs(2), s => s(2), cout => c(3), csout => cs(3) );
  addbit3: add_hibits port map
    (x => x(3), y => y(3), blkcin => cin, ripcin => c(3),
     csin => cs(3), s => s(3), cout => c(4), csout => cs(4) );
  cout <= c(4) when cs(4) = '1' else cin;
end add4_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity add5 is
  port(
    x: in STD_LOGIC_VECTOR(0 to 4);
    y: in STD_LOGIC_VECTOR(0 to 4);
    cin: in STD_LOGIC;
    cout: out STD_LOGIC;
    s: out STD_LOGIC_VECTOR(0 to 4)
  );
end add5;

architecture add5_arch of add5 is
  signal xn: STD_LOGIC_VECTOR(1 to 4);
  signal c, cs: STD_LOGIC_VECTOR(1 to 5);
  component add_01
     port (
```

```
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        cin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        x_xnor_y: out STD_LOGIC
      );
  end component;
  component add_hibits
    port (
        x: in STD_LOGIC;
        y: in STD_LOGIC;
        blkcin: in STD_LOGIC;
        ripcin: in STD_LOGIC;
        csin: in STD_LOGIC;
        s: out STD_LOGIC;
        cout: out STD_LOGIC;
        csout: out STD_LOGIC
    );
  end component;
begin
-- This part read from a file called add_01_instantiation.vhdl
-- component instantiation of adder bits 0 and 1 for block of bits

  addbit0: add_01 port map
    (x => x(0),
     y => y(0),
     cin => cin,
     s => s(0),
     cout => c(1),
     x_xnor_y => cs(1)
    );

  addbit1: add_01 port map
    (x => x(1),
     y => y(1),
     cin => c(1),
     s => s(1),
     cout => c(2),
     x_xnor_y => xn(1)
    );

  cs(2) <= xn(1) or cs(1);


-- component instantiation of higher bits

  addbit2: add_hibits port map
    (x => x(2), y => y(2), blkcin => cin, ripcin => c(2),
     csin => cs(2), s => s(2), cout => c(3), csout => cs(3) );
  addbit3: add_hibits port map
    (x => x(3), y => y(3), blkcin => cin, ripcin => c(3),
     csin => cs(3), s => s(3), cout => c(4), csout => cs(4) );
  addbit4: add_hibits port map
    (x => x(4), y => y(4), blkcin => cin, ripcin => c(4),
     csin => cs(4), s => s(4), cout => c(5), csout => cs(5) );
  cout <= c(5) when cs(5) = '1' else cin;
end add5_arch;

-- TOP LEVEL ENTITY-ARCHITECTURE PAIR

library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity add15 is
  port (
    x, y: in STD_LOGIC_VECTOR (0 to 14);
    cin: in STD_LOGIC;
    s: out STD_LOGIC_VECTOR (0 to 14);
    cout: out STD_LOGIC
  );
end add15;

architecture add15_arch of add15 is
  signal c: STD_LOGIC_VECTOR (1 to  4);
  component add1
    port(
      x: in STD_LOGIC;
      y: in STD_LOGIC;
      cin: in STD_LOGIC;
      cout: out STD_LOGIC;
      s: out STD_LOGIC
    );
  end component;

  component add2
    port(
      x: in STD_LOGIC_VECTOR(0 to 1);
      y: in STD_LOGIC_VECTOR(0 to 1);
      cin: in STD_LOGIC;
      cout: out STD_LOGIC;
      s: out STD_LOGIC_VECTOR(0 to 1)
    );
  end component;

  component add3
    port(
      x: in STD_LOGIC_VECTOR(0 to 2);
      y: in STD_LOGIC_VECTOR(0 to 2);
      cin: in STD_LOGIC;
      cout: out STD_LOGIC;
      s: out STD_LOGIC_VECTOR(0 to 2)
    );
  end component;

  component add4
    port(
      x: in STD_LOGIC_VECTOR(0 to 3);
      y: in STD_LOGIC_VECTOR(0 to 3);
      cin: in STD_LOGIC;
      cout: out STD_LOGIC;
      s: out STD_LOGIC_VECTOR(0 to 3)
    );
  end component;

  component add5
    port(
      x: in STD_LOGIC_VECTOR(0 to 4);
      y: in STD_LOGIC_VECTOR(0 to 4);
      cin: in STD_LOGIC;
      cout: out STD_LOGIC;
      s: out STD_LOGIC_VECTOR(0 to 4)
    );
  end component;

begin
```

```
    blk0: add1 port map
     (x => x(0),
      y => y(0),
      s => s(0),
      cin => cin,
      cout => c(1)
    );
    blk1: add2 port map
     (x => x(1 to 2),
      y => y(1 to 2),
      s => s(1 to 2),
      cin => c(1),
      cout => c(2)
    );
    blk2: add3 port map
     (x => x(3 to 5),
      y => y(3 to 5),
      s => s(3 to 5),
      cin => c(2),
      cout => c(3)
    );
    blk3: add4 port map
     (x => x(6 to 9),
      y => y(6 to 9),
      s => s(6 to 9),
      cin => c(3),
      cout => c(4)
    );
    blk4: add5 port map
     (x => x(10 to 14),
      y => y(10 to 14),
      s => s(10 to 14),
      cin => c(4),
      cout => cout
    );
end add15_arch;
```